

# Android混合脚本测试

百度MTC 洪至远





# 极客时间


重拾极客精神 · 提升技术认知

每天10分钟,邀请顶级技术专家,为你传道授业解惑。



扫一扫,试读专栏



主办方 **Geekbang**  **InfoQ**  
极客邦科技

# ArchSummit

全球架构师峰会 2017

12月8-9日 北京·国际会议中心





APSEC 2017



Geekbang  
极客邦科技

InfoQ

# APSEC 2017

24th Asia-Pacific Software Engineering Conference  
4-8 December 2017, Nanjing, Jiangsu, China

12月4-8日

中国南京



了解详情



# AiCon

全球人工智能技术大会 2018

## 助力人工智能落地

2018.1.13 - 1.14 北京国际会议中心



扫描关注大会官网



# 概览

- 问题背景
- 如何解决脚本执行的干扰
- 如何支持各类场景的脚本测试
- 如何快速生成脚本

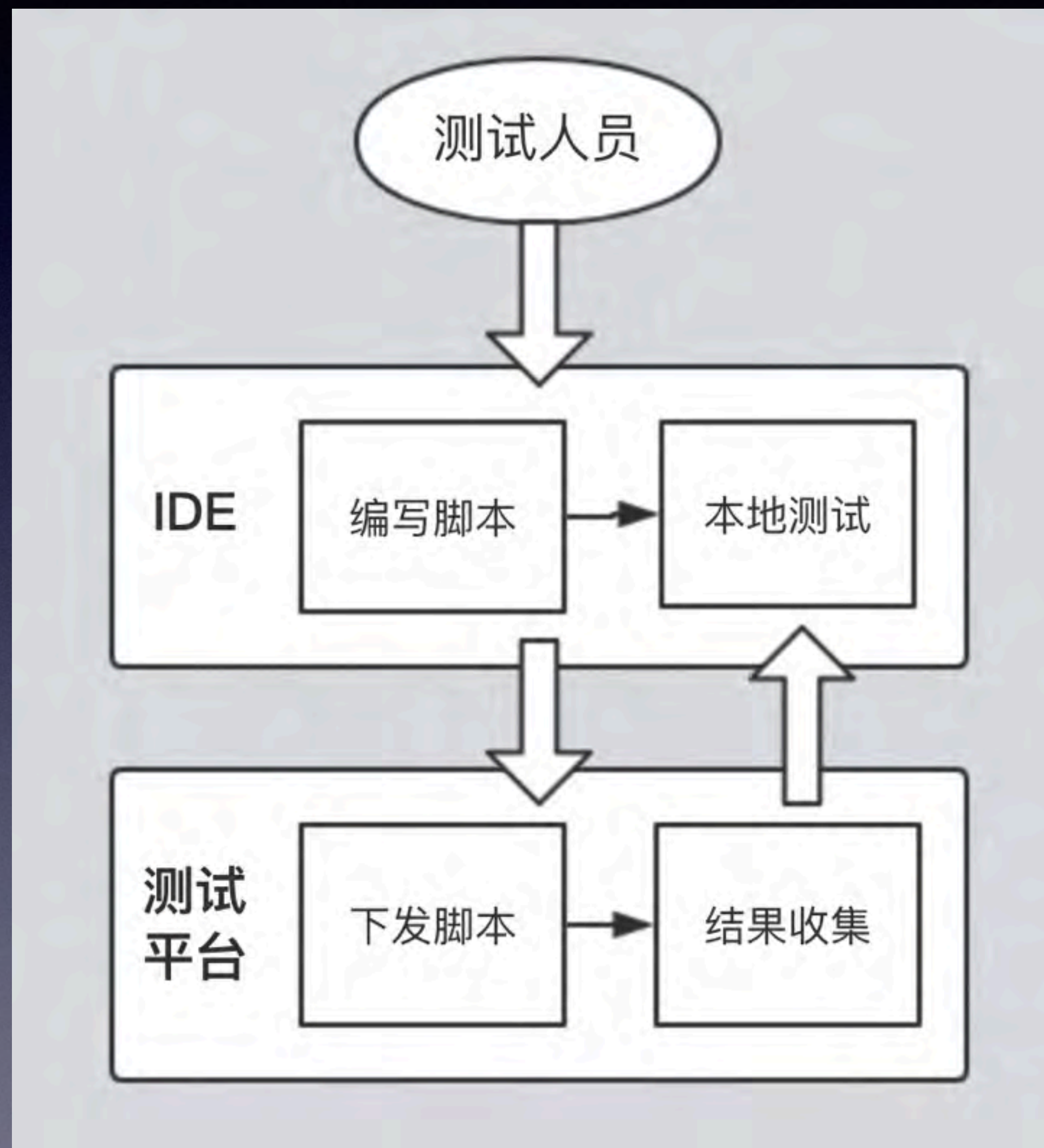


# 为什么需要自动化测试

- 缩短测试周期
- 统一标准，避免人为出错
- 提升测试覆盖率

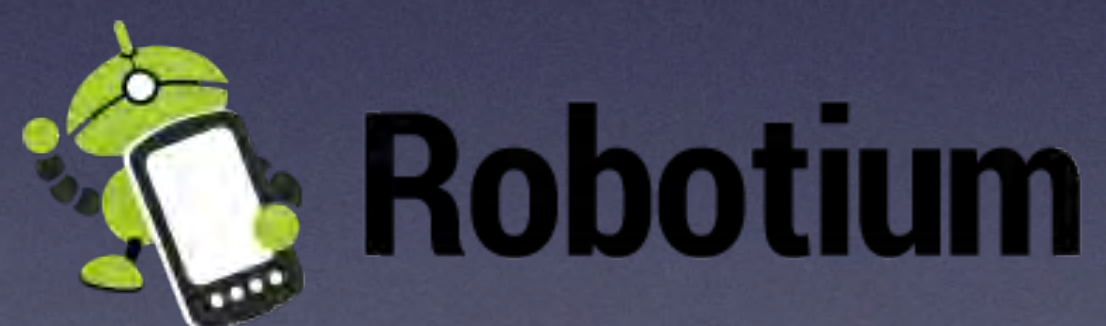


# 常规的自动化测试场景





# 如何选择测试框架?





# 常用框架比较

测试框架	Robotium	Appium	UIAutomator
优点	<div><div>✓ 可以进行黑盒测试</div><div>✓ 基于插桩机制实现，对非原生应用支持较好</div></div>	<div><div>✓ 支持跨平台（Android/iOS）</div><div>✓ 针对不同Android版本有不同支持方案，可扩展</div></div>	<div><div>✓ 非侵入式机制</div><div>✓ 支持跨进程界面获取</div></div>
缺点	<div><div>✓ 不支持跨进程</div><div>✓ 需要对待测APP进行重签名，对加壳的APP重签名后会运行失败</div></div>	<div><div>✓ 稳定性不足</div></div>	<div><div>✓ 不支持Android4.1以下系统</div><div>✓ 对非原生界面支持较差</div></div>



# 仍然面对的问题

- 脚本执行容易受到干扰
- 非Native场景脚本支持较差
- 脚本编写比较耗时，如何提升效率

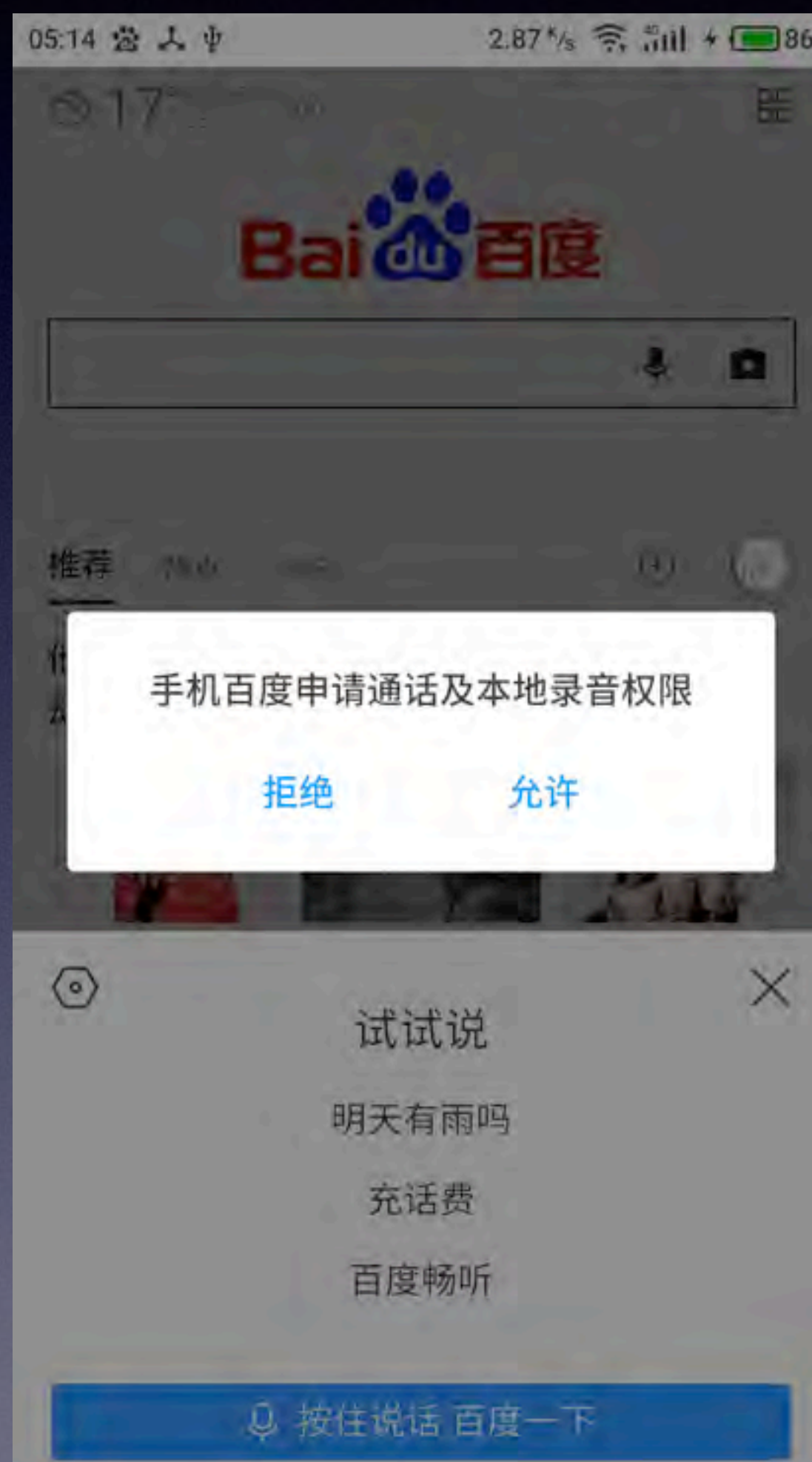


# 仍然面对的问题

- 脚本执行容易受到干扰
- 非Native场景脚本支持较差
- 脚本编写比较耗时，如何提升效率



# 脚本测试中有哪些干扰?





# 什么是弹窗干扰

进程外弹窗



+

进程内弹窗



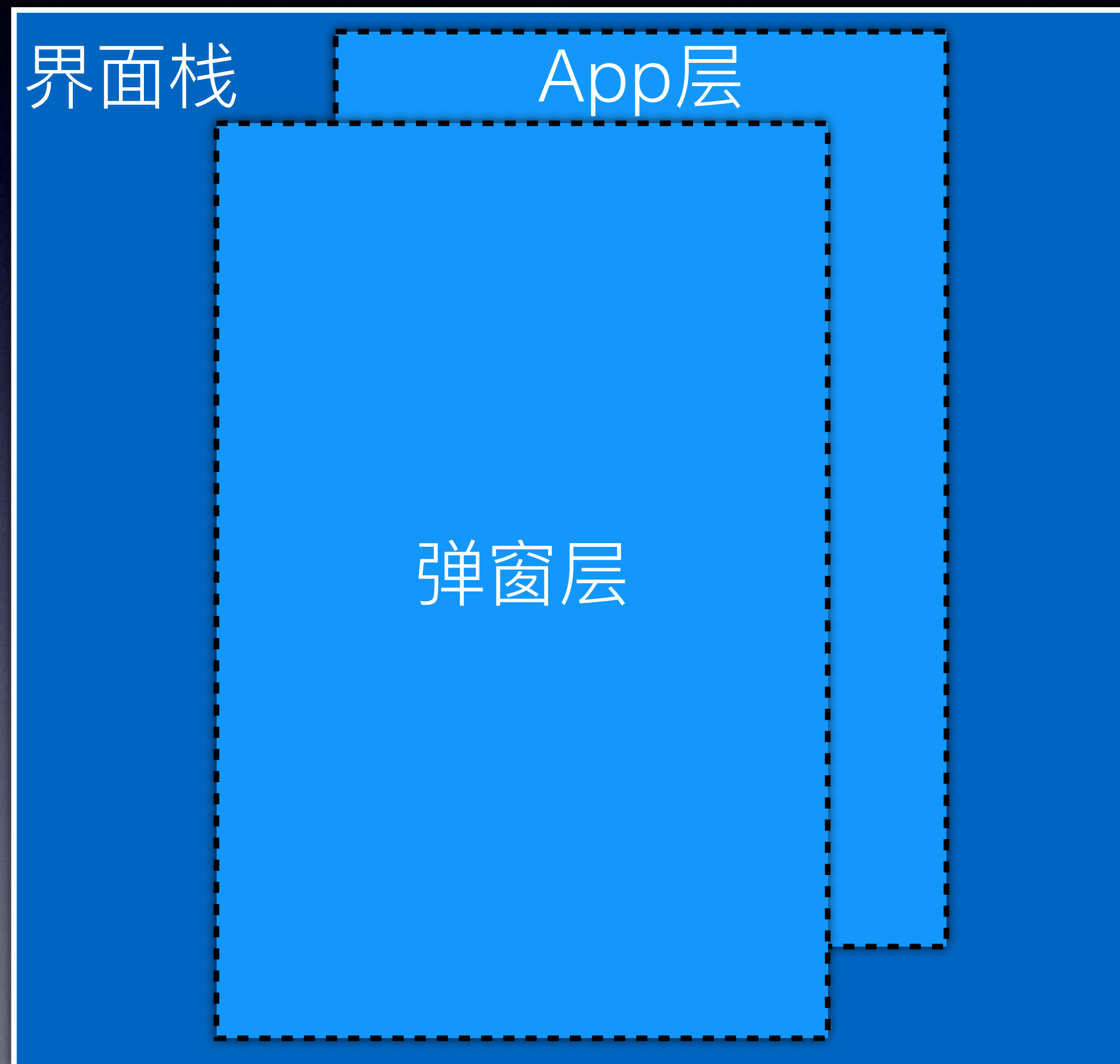


# 如何解决弹窗干扰

- 进程内弹窗可通过脚本逻辑清理掉
- UiAutomator支持跨进程操作，可处理进程外弹窗



# 如何发现弹窗





# 定位弹窗

## 方案一：基于元素

- 从根本解决问题
- 需要根据系统适配

## 方案二：基于文本

- 方法比较简单
- 需要适配场景

综合考虑后，我们采取了基于文本的方案



# 扩展UIAutomator—清道夫SDK

- 基于UIAutomator的API进行封装
- 操作时自动清理弹窗
- 增加操作重试和记忆功能，保证操作的正常执行
- 增加自定义截图功能支持点击位置展示
- ...

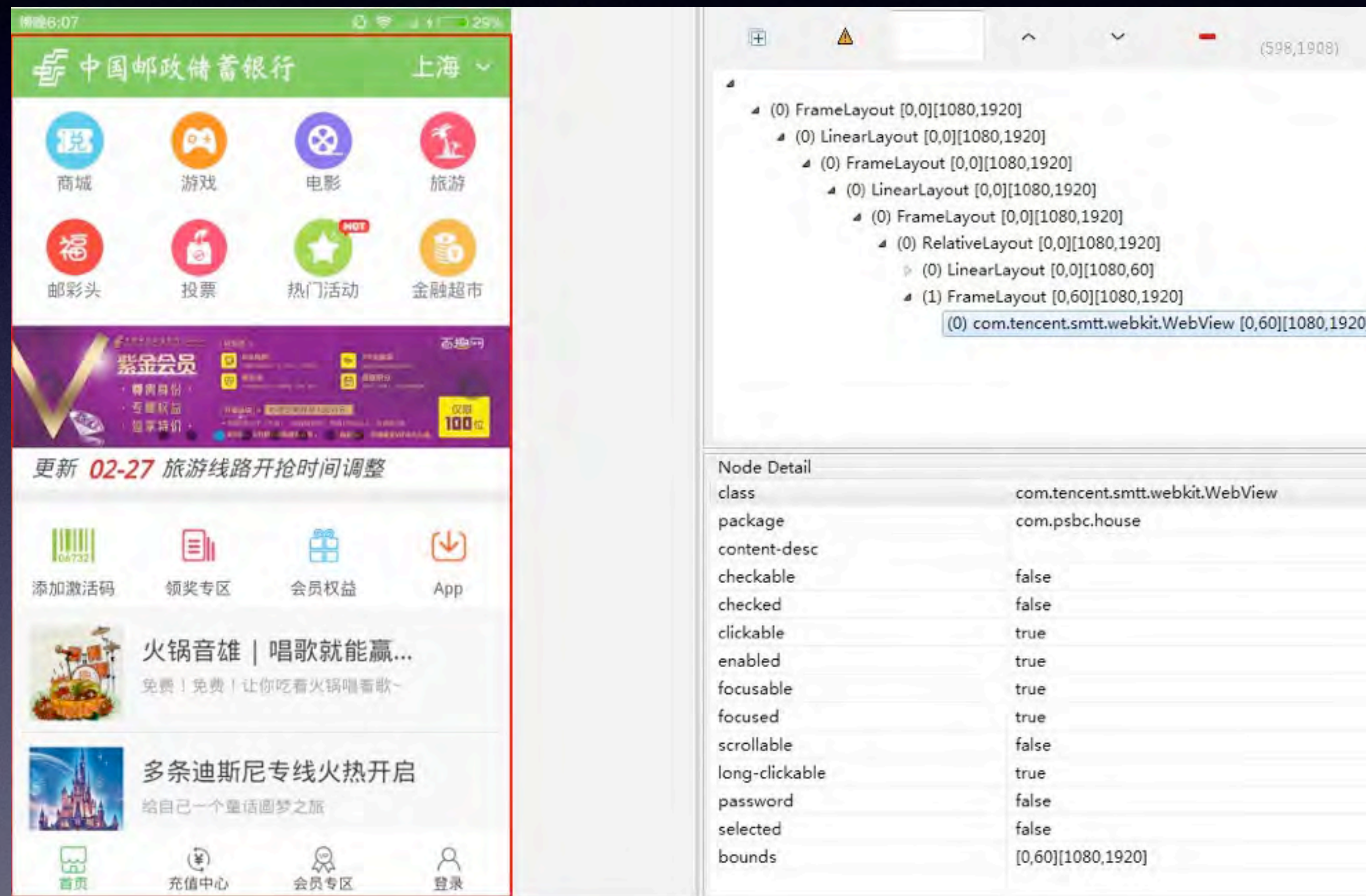


# 仍然面对的问题

- 脚本执行容易受到干扰
- 非Native场景脚本支持较差
- 脚本编写比较耗时，如何提升效率

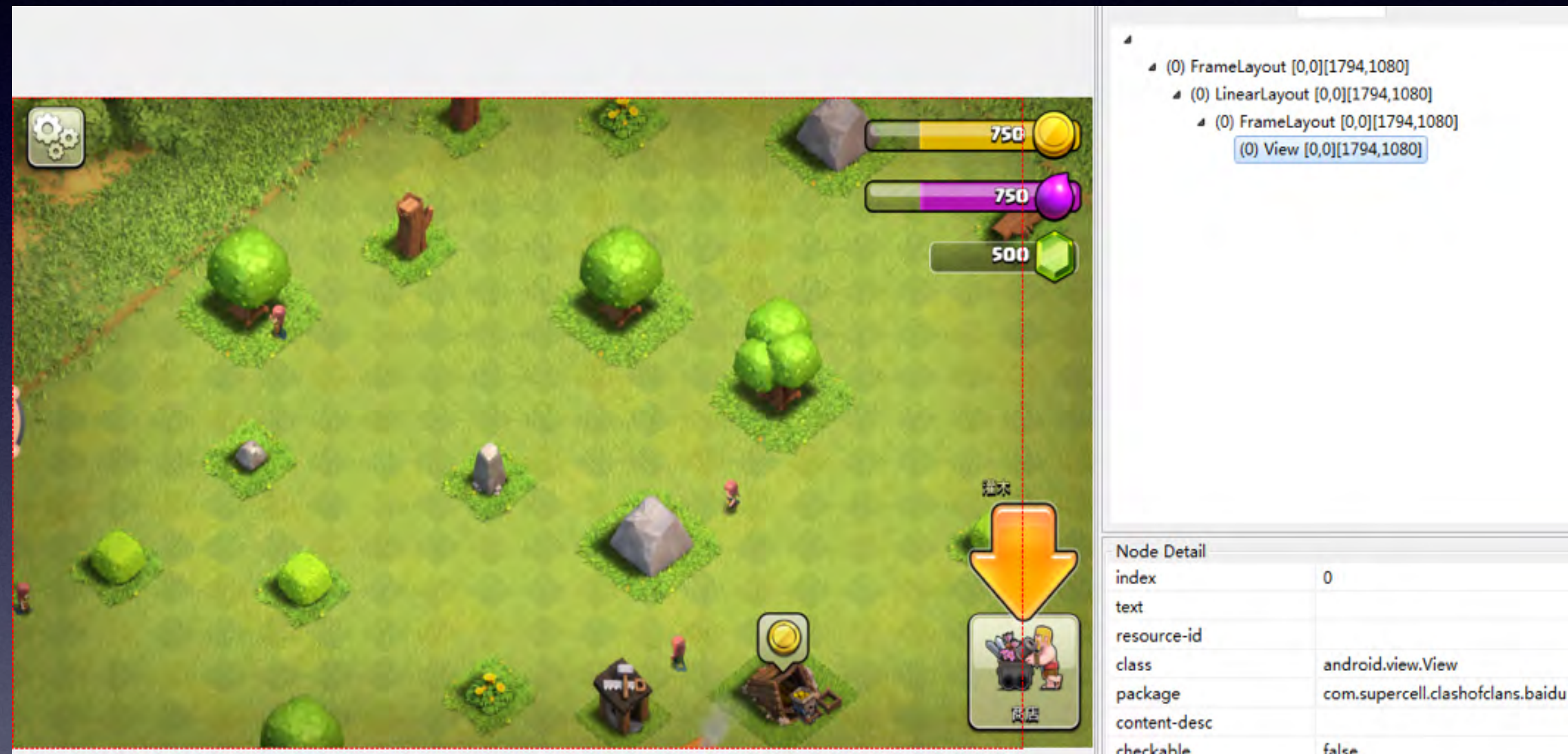


# 问题场景1：WebView



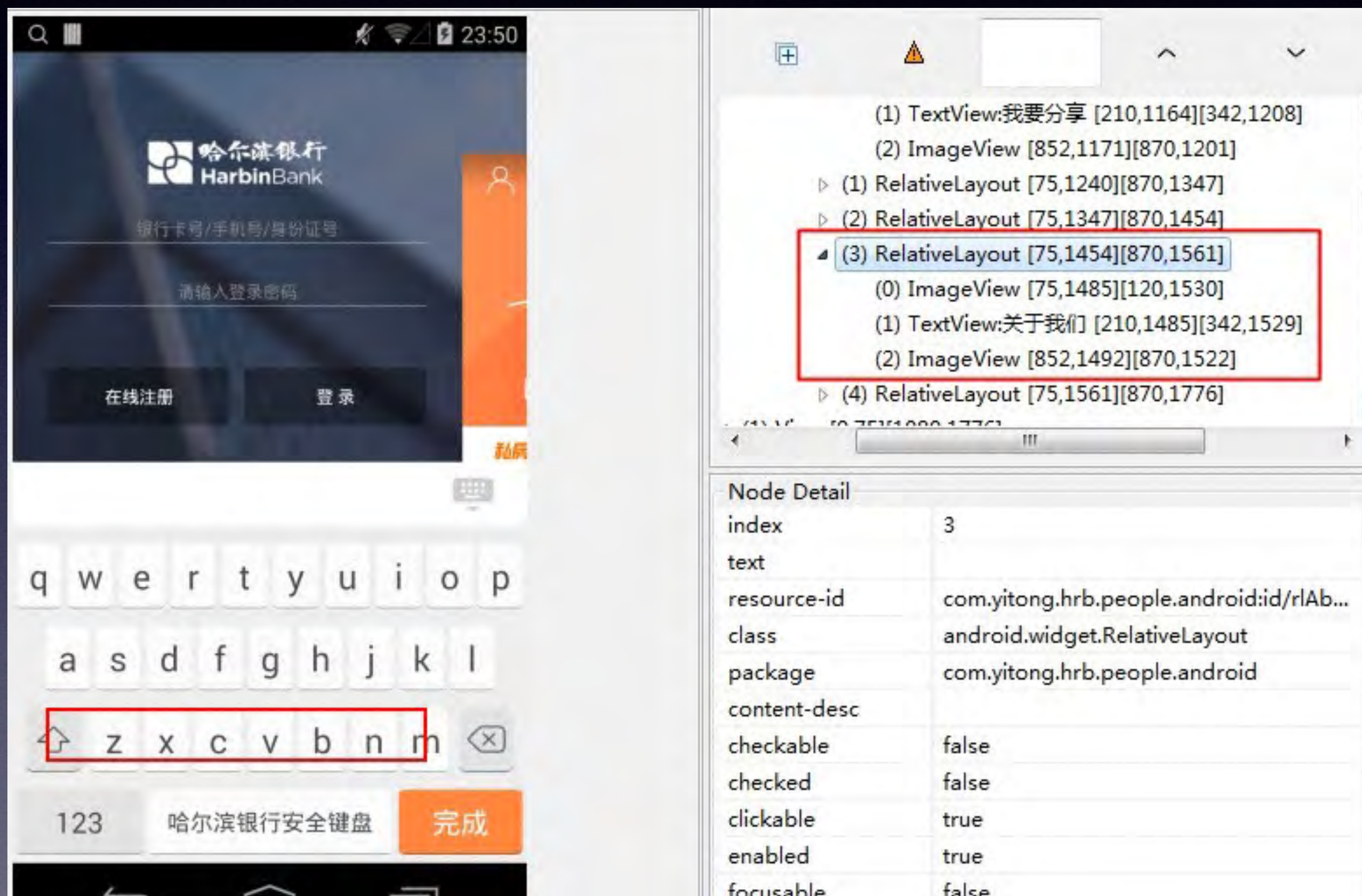


# 问题场景2：游戏





# 问题场景3：PopupWindow



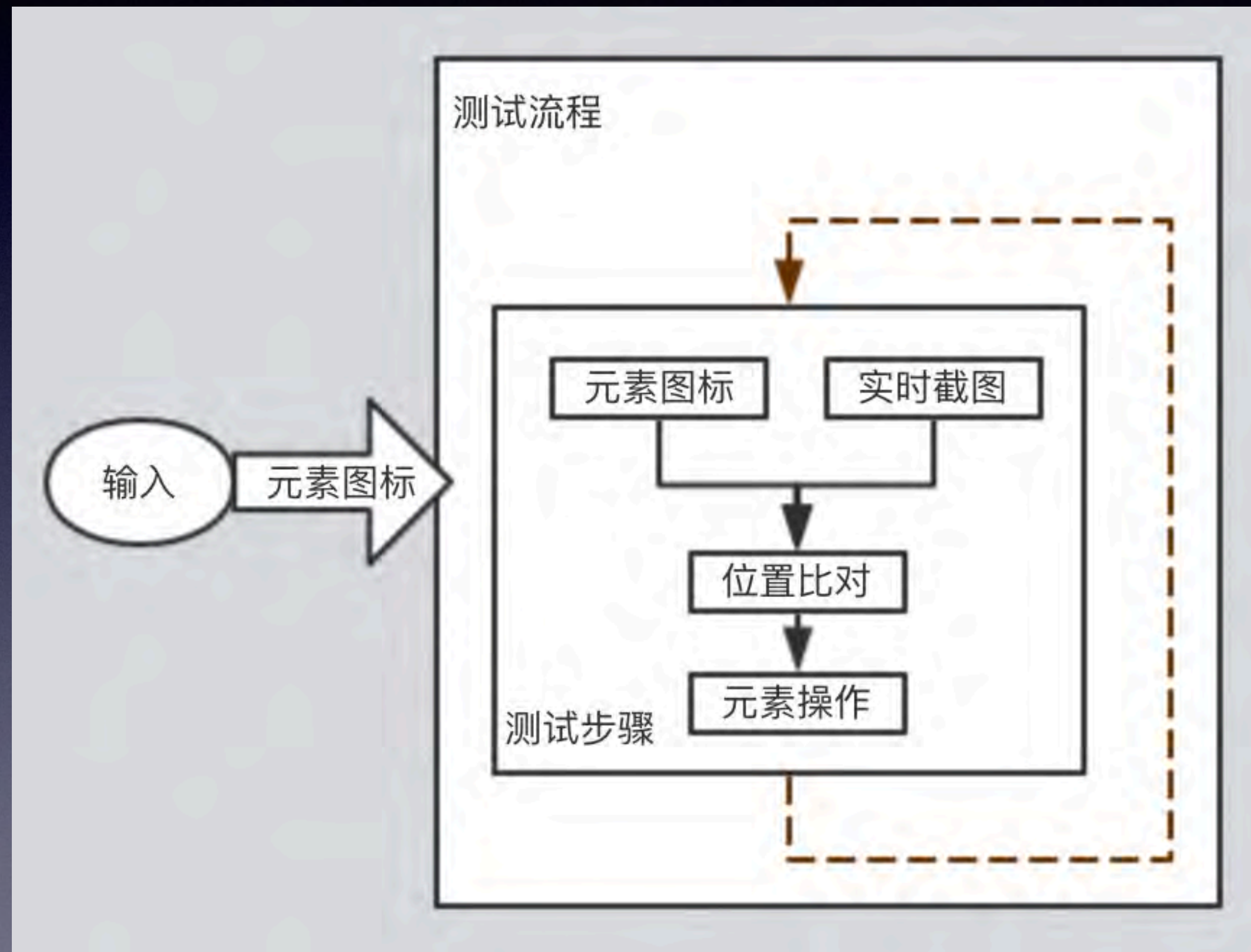


# 可行的解决方案——基于元素

- WebView通过参考Appium的原理基于ChromeDriver来进行元素解析
- 游戏通过嵌入SDK，针对引擎特点反射UI控件元素
- PopupWindow通过代码层增加focus操作



# 可行的解决方案—图像识别

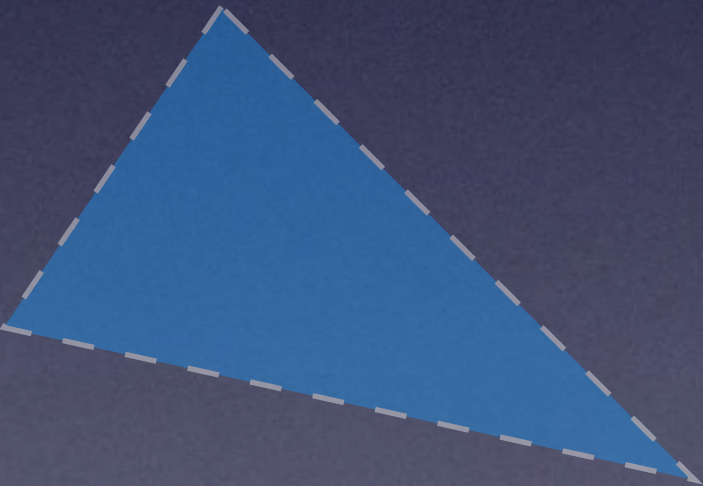




# 方案对比

基于元素的方案

稳定性

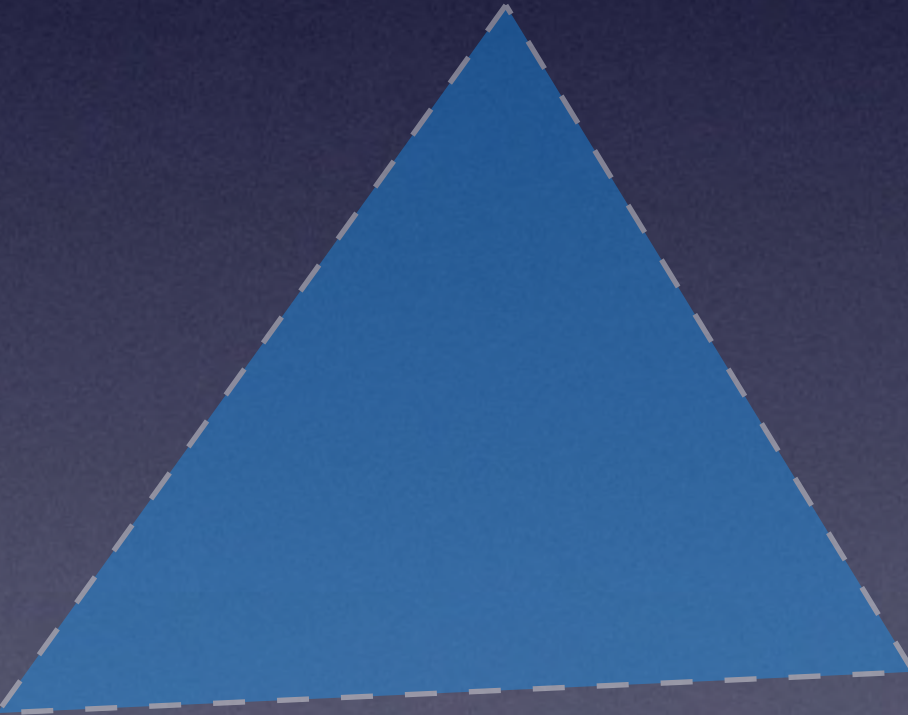


易用性

准确性 易用性

基于图像识别的方案

稳定性



准确性



# 图像识别的匹配算法

## 模板匹配算法

### 优势

速度快  
算法简单

### 劣势

容易受干扰

## 特征点匹配算法

### 优势

精度高  
抗干扰能力强

### 劣势

速度慢  
简单图形匹配精度不高



# 基于OCR的方案

- 基于OCR可以提升对文字、字母等简单图形等识别准确率





# 工程化解决方案

- 整合三种方案（模板算法、特征点算法、OCR)
- 增加不同分辨率下截图，并自动匹配
- 限定匹配区域，提高匹配准确度



# 元素识别 vs 图像识别

## UIAutomator

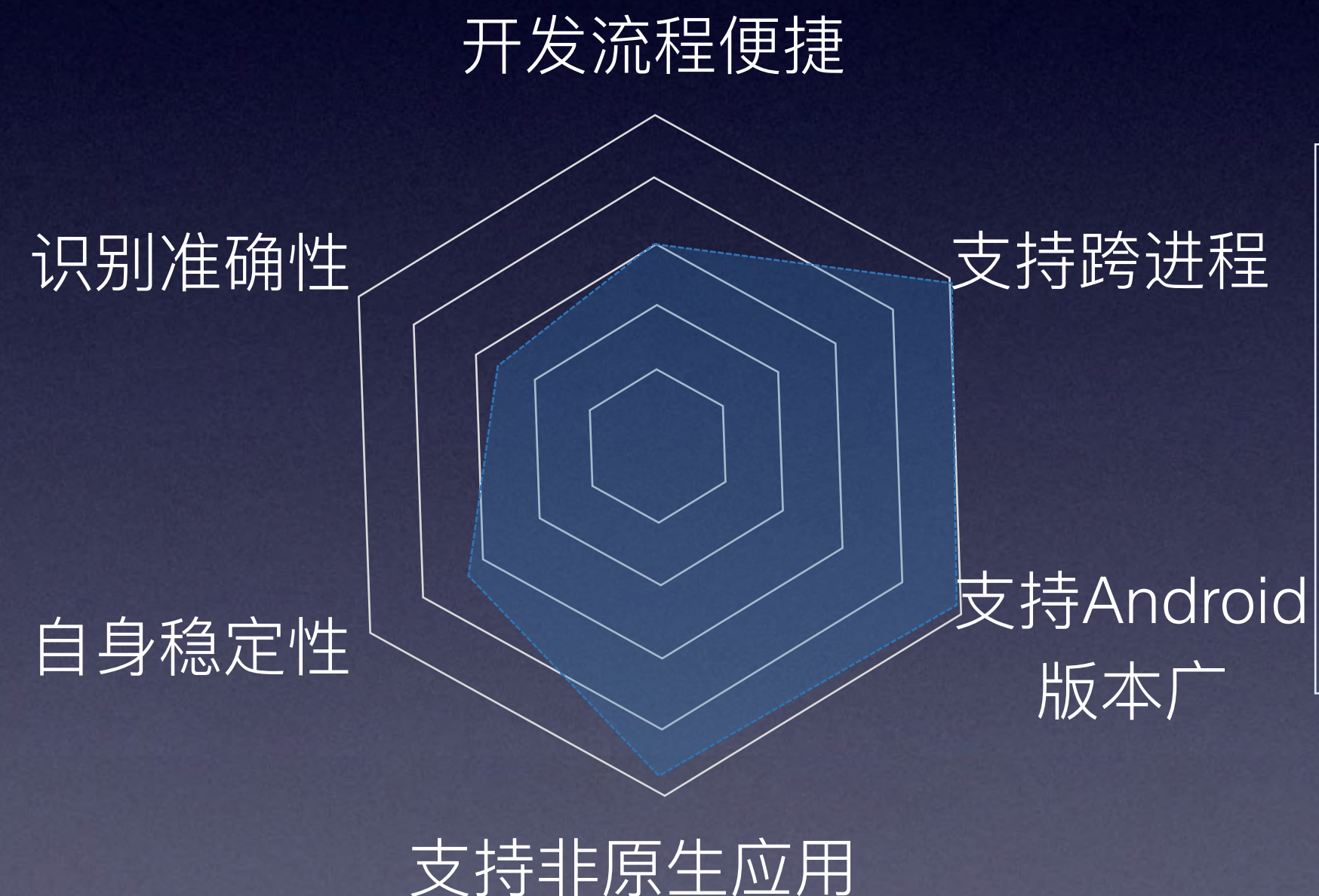
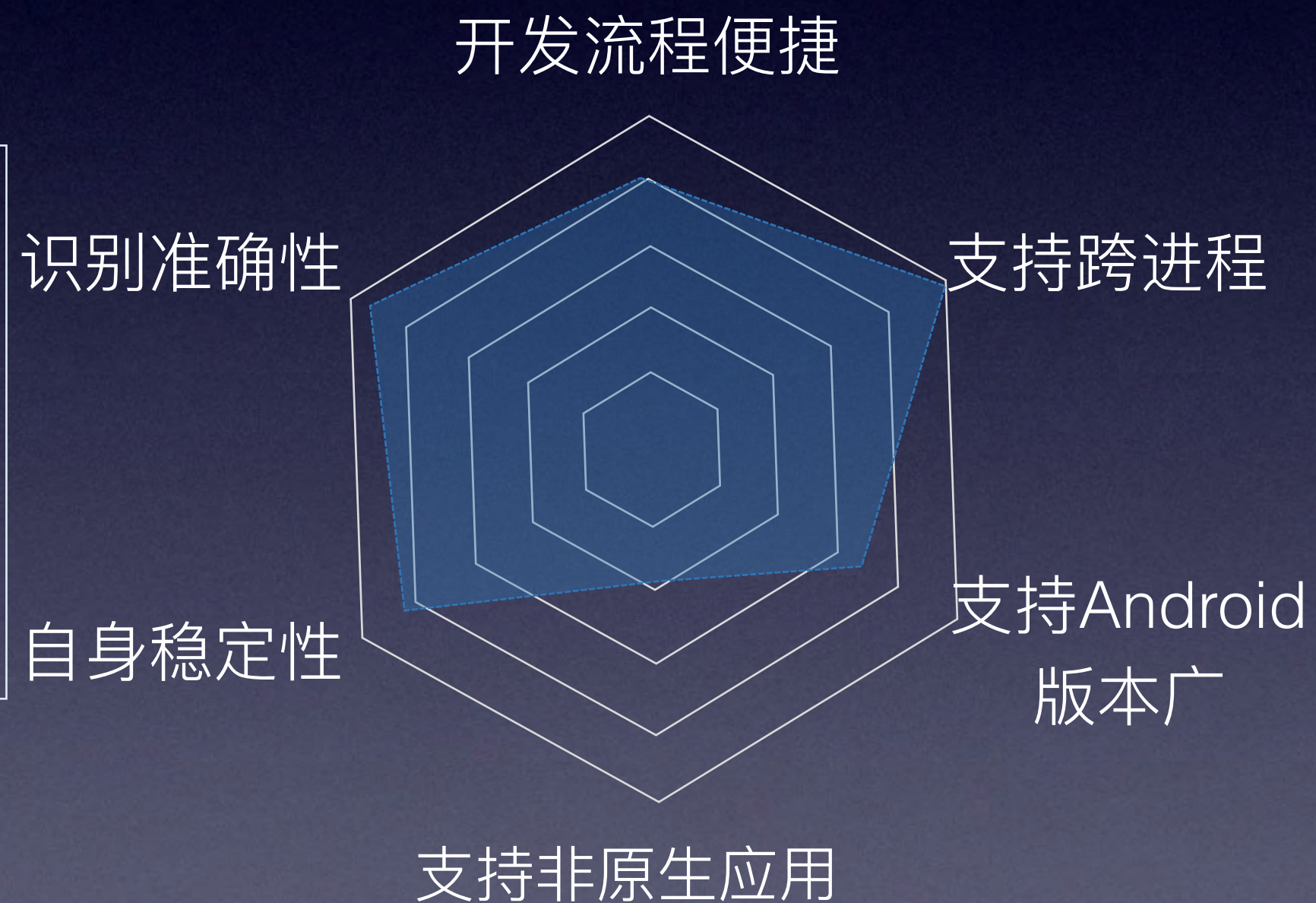
## 图像识别测试

### 能力标签

基于控件

亲儿子出品

擅长原生应用



### 能力标签

基于图像匹配

通用成熟算法

擅长非原生应用

控件识别元素 和 图像识别元素 各有优缺点，且互为补充

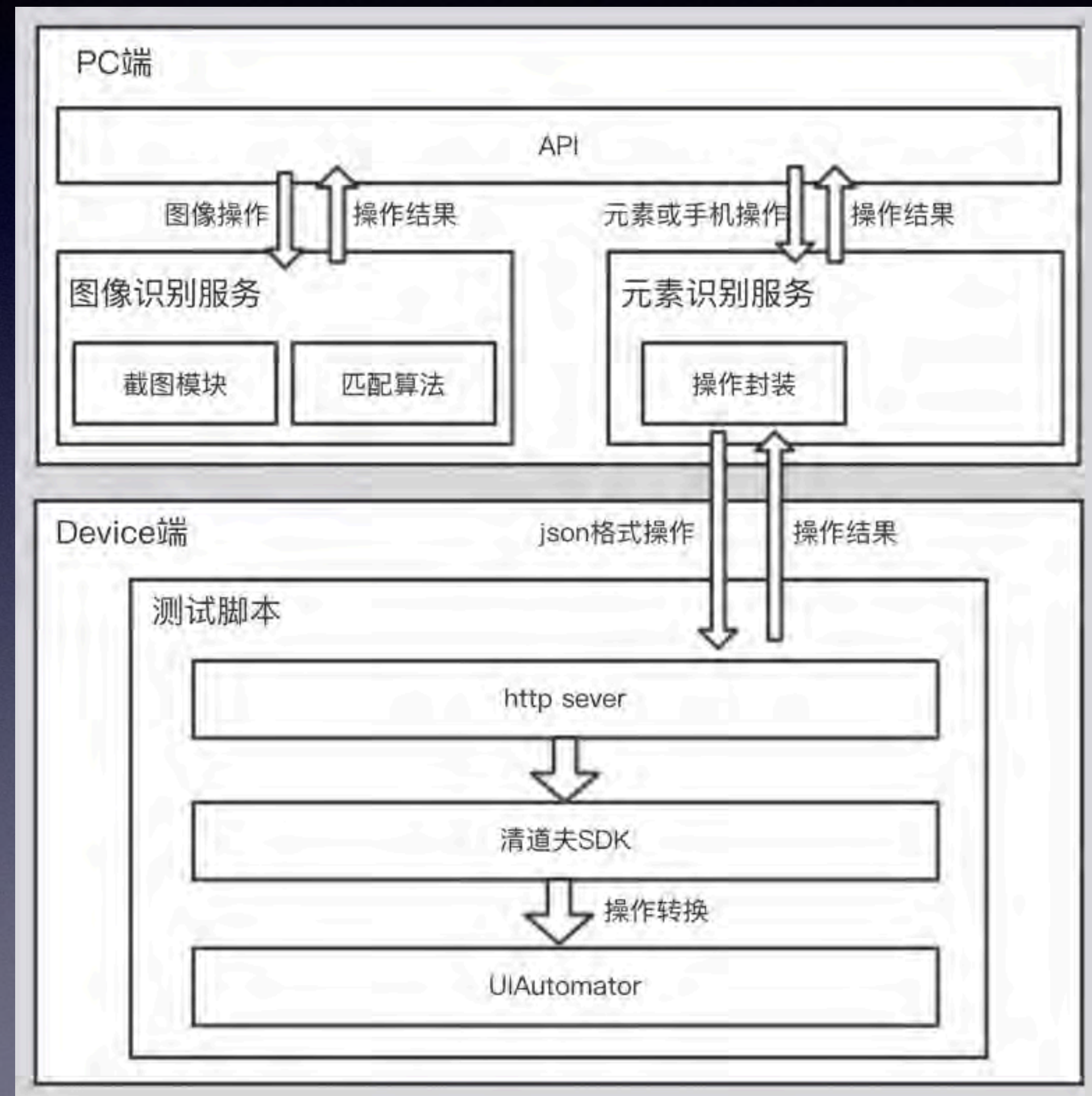


# 如何结合两种方案

- 将图像识别能力迁移至手机端（可能会影响测试）
- 将UIAutomator迁移至PC端（有成熟解决方案）



# 混合脚本测试解决方案





# 混合脚本测试解决方案

- 对UIAutomator的封装，提升了脚本的稳定性
- 综合使用各种图像识别技术，极大的覆盖了非Native场景下的测试
- 整合API，使得通过一套脚本即可覆盖各类场景

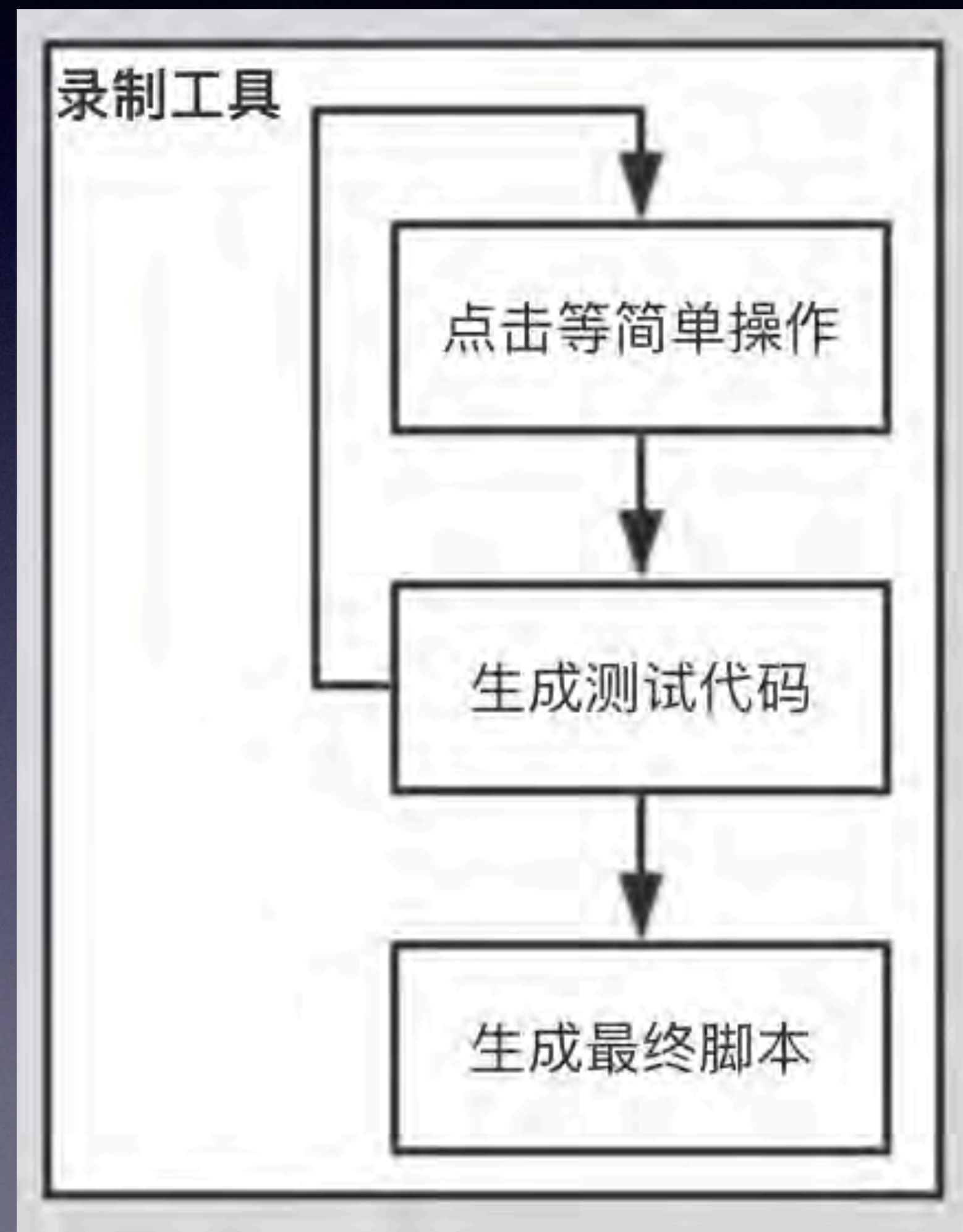


# 仍然面对的问题

- 脚本执行容易受到干扰
- 非Native场景脚本支持较差
- 脚本编写比较耗时，如何提升效率



# 测试脚本录制工具





# 核心功能

- 手机屏幕展示
- 手机操作功能
- 根据操作生成脚本 **(难点)**



# 现有的脚本生成方案

## 基于坐标的方案

### 优势

简单

### 劣势

受分辨率影响太大

## 基于元素属性的方案

### 优势

简单

通用性较好

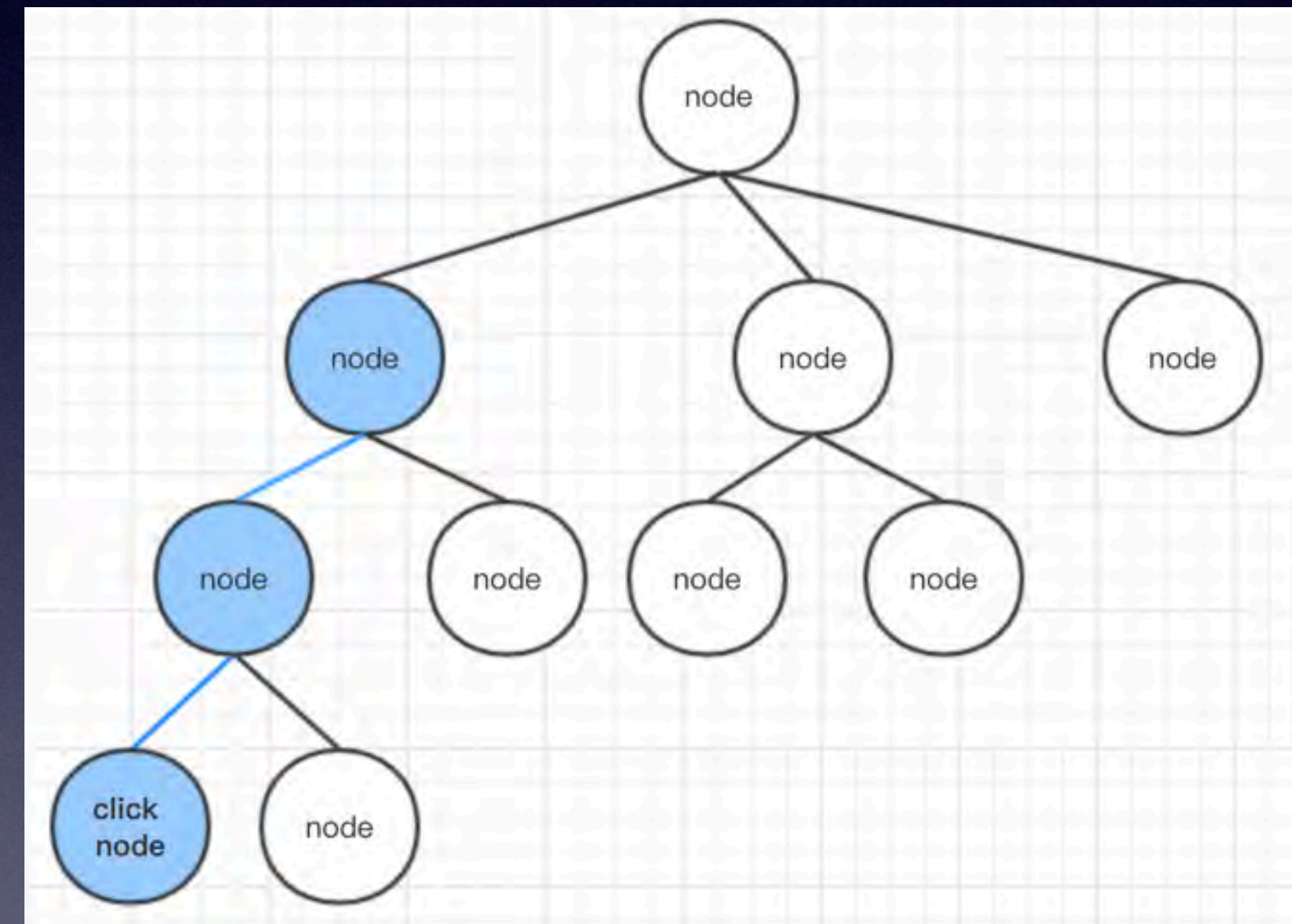
### 劣势

元素属性可能不唯一



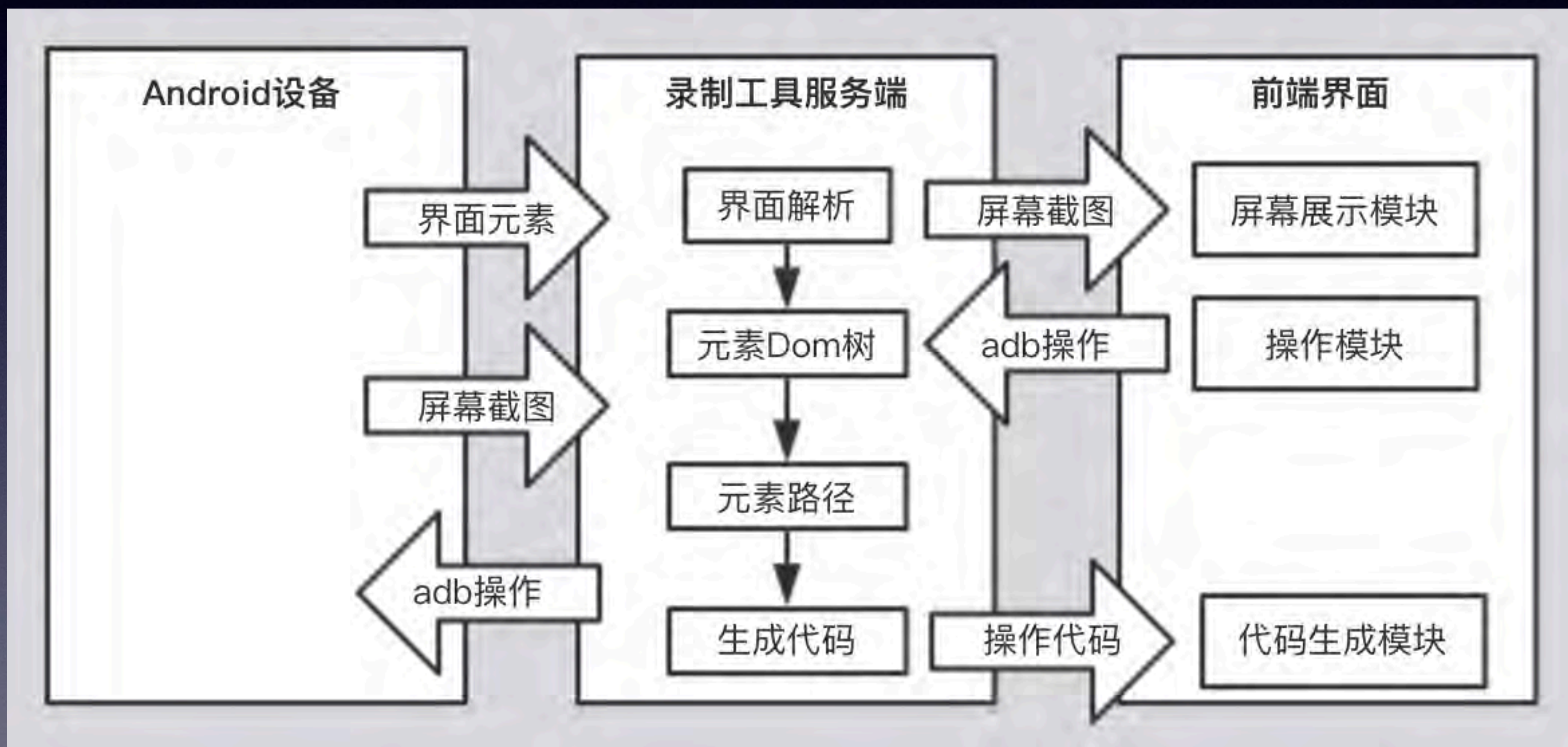
# 基于点击路径的方案

- 构造页面的元素树结构
- 点击时记录往根节点回溯路径，综合每个节点的各项属性，唯一定位当前点击元素





# 录制工具整体解决方案





# 整体自动化测试解决方案

- 基于录制工具录制脚本
- 在MTC平台下发任务
- 查看测试报告



# 参考资料

- <http://appium.io/>
- <https://robotium.com/>
- <https://developer.android.com/training/testing/ui-automator.html>
- [https://en.wikipedia.org/wiki/Scale-invariant\\_feature\\_transform](https://en.wikipedia.org/wiki/Scale-invariant_feature_transform)
- [https://en.wikipedia.org/wiki/Speeded\\_up\\_robust\\_features](https://en.wikipedia.org/wiki/Speeded_up_robust_features)
- <http://opencv-python-tutroals.readthedocs.io/en/latest/>
- <https://github.com/openstf/minicap>



Q ? A : End